Working Session 2 – Self-Adaptation on a Body Sensor Network to mitigate false-positives

Teaching Assistants: Ricardo D. Caldas, Carlos Eduardo, Vicente
Do you have a question? Your TA is taking too long to answer? Post it on Telegram!

Problem: The Body Sensor Network (BSN) was developed to calculate whether a patient is in a risk state which will trigger an emergency alert to medical doctors. However, the current implementation does not account for varying, and unexpected, operational contexts. This leads to false-positive emergency alerts which affect trust from users in the product. For example, in Figure 1, the current implementation considers that a value of 138 bpm in the heart rate measurements pose a high risk to the patient, which may trigger an alert to medical doctors. Although such a risk estimate is correct for a patient who is sitting in the park (context 1), it is not correct for a patient running on the treadmill. When running, humans are expected to have average values of 100-150 bpm. In fact, the BSN returns 36% of false-positives for the signal in Figure 1.

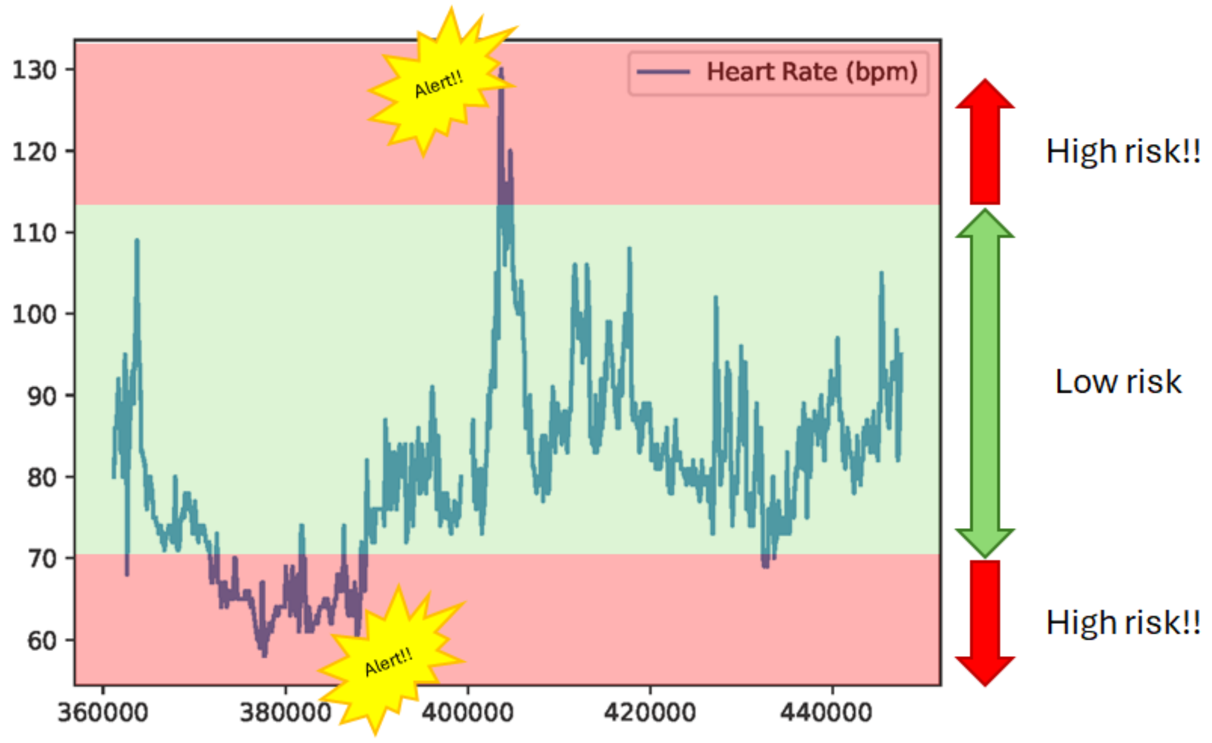


Figure 1. Heart beat rate of a patient from 360000s to 440000s.

*Your **objective** is to implement an algorithm that infers the context in which the patient is emerged in (i.e., sitting, running, or sleeping) and adapts the expected risk levels.*

A traditional framework to implement self-adaptation uses four activities:
Monitoring, Analysis, Planning, and Execution.

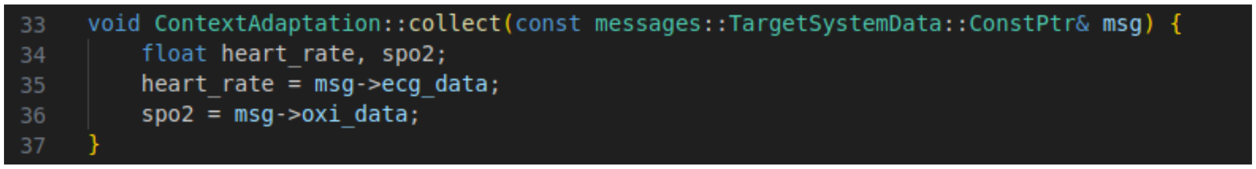
We designed a set of steps to help you achieve that:

Step 0. You must define the different contexts and how they affect the risk levels.

Sensor	**Sitting**	**Running**	**Sleeping**
Heart Beat Rate	Low risk : [85,97] Mod. risk: [70,85] [97,115] High risk: [0,70] [115,300]	Low risk : [ ] Mod. risk: [ ] [ ] High risk: [ ] [ ]	Low risk : [ ] Mod. risk: [ ] [ ] High risk: [ ] [ ]
SpO2	Low risk : [65,100] Mod. risk: [55,65] [-1,-1] High risk: [0,55] [-1,-1]	Low risk : [ ] Mod. risk: [ ] [ ] High risk: [ ] [ ]	Low risk : [ ] Mod. risk: [ ] [ ] High risk: [ ] [ ]

Step 1. [Monitoring] Then, gather runtime information from a vital sign.

There is a mockup node with a simple callback implemented for reading the sensor's data from the BSN execution.



```
void ContextAdaptation::collect(const messages::TargetSystemData::ConstPtr& msg) {
    float heart_rate, spo2;
    heart_rate = msg->ecg_data;
    spo2 = msg->oxi_data;
}
```

In this node the function collect() receives the sensor data and separates the spo2 and in each variable. For instance, if you would like to monitor the heart beat rate, you can pass the values to arrays which will be later used for analysis.

Step 2. [Analysis] With the information from the vital signs in hands, one must analyze and understand what should trigger a change of context.

This step asks for understanding whether there is a change in context. For example, whether the person using the BSN is not sitting anymore but running, or sleeping. To do this, you you can calculate a moving average from the last readings (stored in the array) and its changes could determine a context change.

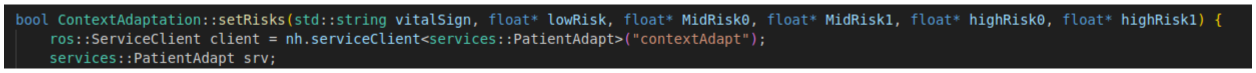
Step 3. [Planning] Given the change of context this step should transform the new information of context into a series of steps to change the BSN in order to change the risk levels, according to Table 1.

This step should look like a switch-case structure:
if context is sitting:

```
    set_risk(vitalSign="heart_rate",low_risk=[85,97],mod_risk=[97,105],high_risk=[105,200 ])
else if context is sleeping:
    set_risk(vitalSign="heart_rate",low_risk=[60,85],mod_risk=[85,97],high_risk=[97,150])
else if context is running:
    …
```

Step 4. [Execution] Once there is a series of actions defined to change the BSN, this step should emit the changes

In the mockup node there is also a function that changes the risk values for oxygenation and heart rate. The setRisks(...) receives a string that must contain "heart_rate" or "oxigenation" for change the params for each sensor, otherwise it won't work, and a series with five array floats, each one must contain two values indicating the floor and the ceil for each state; respectively LowRisk, MidRisk0, MidRisk1, HighRisk0, HighRisk1.



```
bool ContextAdaptation::setRisks(std::string vitalSign, float* lowRisk, float* MidRisk0, float* MidRisk1, float* highRisk0, float* highRisk1) {
    ros::ServiceClient client = nh.serviceClient<services::PatientAdapt>("contextAdapt");
    services::PatientAdapt srv;
```

Challenge:

Fork a repository from the BSN].
Update the code accordingly to your solution.

1) Describe your solution in the ReadMe.md, motivating your decisions.
2) Also, upload the log from a successful execution.

Submit your solution here: [link to google forms]
Submissions will be accepted until 11:59pm on Thursday 22th of Feb.